Chernoff Bounds

1 Application of Chernoff Bounds

Chernoff Bounds

\[ Pr[|p - \hat{p}| > \alpha] < 2e^{-2n\alpha^2} \]  
(1)

where \( X_1, \ldots, X_N \) are IID Bernoulli variables with \( E[X_i] = Pr[X_i = 1] = p \)

\[ \hat{p} = \frac{1}{N} \sum X_i \]  
(2)

- Chernoff’s inequality can be used to calculate how large \( n \) needs to be in order to achieve confidence \((1 - \delta)\) in our estimate.

\[
Pr[|p - \hat{p}| > \alpha] < \delta \\
\Leftrightarrow 2e^{-2n\alpha^2} < \delta \\
\Leftrightarrow e^{-2n\alpha^2} < \frac{\delta}{2} \\
\Leftrightarrow -2n\alpha^2 < \ln \frac{\delta}{2} \\
\Leftrightarrow 2n\alpha^2 \geq \ln \frac{2}{\delta} \\
\Leftrightarrow n \geq \left( \frac{1}{2\alpha^2 \ln \frac{2}{\delta}} \right)
\]

Example 1

Consider two coins: coin1 with \( p_1 = \frac{1}{2} \) and coin2 with \( p_2 = \frac{1}{4} \). We do not know which coin is which. Design an experiment to pick coin1.

- **Algorithm:**
  Pick any coin, flip it \( n \) times and calculate \( \hat{p} \). If \( \hat{p} \geq \frac{3}{8} \) then this is coin1, otherwise it is coin2.

  We claim that with \( Pr > 1 - \delta \) the algorithm picks the coin correctly. There are two (symmetric) cases depending on which coin is being flipped by the algorithm. Assume that the algorithm is using coin1, then to make a mistake we have, \( p_1 = \frac{1}{2} \) and \( \hat{p}_1 \geq \frac{3}{8} \) therefore \( \alpha = \frac{1}{8} \).
Applying Chernoff’s inequality:

\[ \Pr[|p_1 - \hat{p}_1| > \frac{1}{8}] < 2e^{-2n\frac{1}{8}} \]
\[ 2e^{-\frac{n}{8}} < \delta \]
\[ n > 32\ln \frac{2}{\delta} \]

The case when the algorithm uses coin2 is similar. Thus, regardless of the coin being used by the algorithm, with probability \( \geq (1-\delta) \) coin1 can be distinguished correctly.

**Example 1b**

Consider two coins: coin1 with \( p_1 = \frac{1}{2} \) and coin2 with \( p_2 = ? \). We do not know which coin is which. Design an experiment to pick a coin with \( p_i \leq \frac{1}{2} \).

- There exists no solution for this problem, as we cannot know \( p_2 \) in advance. Since there is no guaranteed gap between the two probabilities \( p_1 \) and \( p_2 \), they cannot be distinguished apart.

**Example 2**

Consider \( k \) coins with \( \Pr[\text{Head}] = p_i \) for each coin \( i \). \( p_1 \leq \alpha \). We do not know anything about \( p_2, p_3, \ldots, p_k \). Design an experiment to pick a coin with \( p_i \leq 2\alpha \). Notice that we allow a gap between the guaranteed \( p_1 \leq \alpha \) and required \( \leq 2\alpha \).

- **Algorithm:**
  Flip each coin \( n \) times and calculate \( \hat{p}_i \) for each coin. Pick the coin with minimum \( \hat{p}_i \)

We want to guarantee that any coin with \( p_i > 2\alpha \) is not chosen. This is implied if all \( \hat{p}_i \) are within \( \frac{\alpha}{2} \) of \( p_i \):

\[ \hat{p}_1 < \alpha + \frac{\alpha}{2} \]
\[ \hat{p}_i > 2\alpha - \frac{\alpha}{2} \]

Thus under this condition coin1 is preferred over any coin \( i \) with \( p_i > 2\alpha \). Therefore, if the algorithm fails then at least one of \( |p_i - \hat{p}_i| \) and \( |p_1 - \hat{p}_1| \) is \( > \frac{\alpha}{2} \).

**Theorem:** When using \( n > \frac{2}{\alpha^2} \ln(\frac{2k}{\delta}) \), the algorithm which returns the minimum \( \hat{p} \) picks a good coin (which has \( p_i \leq 2\alpha \)) with probability \( \geq (1-\delta) \).

**Proof:** For each coin,

\[ \Pr[|p_i - \hat{p}_i| > \frac{\alpha}{2}] < 2e^{-2n\frac{\alpha^2}{4}} < \frac{\delta}{k} \]

where the last inequality is implied if

\[ 2e^{-2n\frac{\alpha^2}{4}} < \frac{\delta}{k} \]
\[ -2n\frac{\alpha^2}{4} < \ln\left(\frac{\delta}{2k}\right) \]
\[ n > \frac{2}{\alpha^2} \ln\left(\frac{2k}{\delta}\right) \]
By union bound this implies,

$$Pr[\exists i \text{ s.t. } |p_i - \hat{p}_i| > \frac{\alpha}{2}] \leq k \frac{\delta}{k} = \delta$$

**Randomized Polynomial Time Algorithms**

RP is the set of decision problems that have the following kind of algorithm,

- If input $\in$ No $\Rightarrow$ Always say No.
- If input $\in$ Yes $\Rightarrow$ with probability $\geq \frac{3}{4}$ say Yes.

Can we boost the confidence of an RP algorithm? i.e. replace $\frac{3}{4}$ with $1 - \delta$. The confidence of an RP algorithm can be boosted by running it $k$ times and,

- if the algorithm says Yes at least once $\Rightarrow$ Say yes.
- otherwise say No.

The number of trials needed to achieve the desired confidence can be computed as follows,

$$Pr[\text{algorithm says No } k \text{ times}] \leq \frac{1}{4} < \delta$$

$$\frac{1^{2k}}{2} < \delta$$

$$k > \frac{1}{2} \lg_2 \frac{1}{\delta}$$

**Bounded Probabilistic Polynomial Time Algorithms**

BPP is the set of decision problems that have the following kind of algorithm,

- If input $\in$ No $\Rightarrow$ with probability $\geq \frac{3}{4}$ say No.
- If input $\in$ Yes $\Rightarrow$ with probability $\geq \frac{3}{4}$ say Yes.

Can we boost the confidence of BPP algorithms? i.e. replace $\frac{3}{4}$ with $1 - \delta$. The confidence of a BPP algorithm can be boosted by running it $k$ times and calculating $\hat{p} = \text{fraction of times the algorithm said Yes}$,

- if $\hat{p} \geq \frac{1}{2}$ say Yes.
- otherwise say No.

The output would be wrong only when $|p - \hat{p}_i| > \frac{1}{4}$, this can be used to calculate the number of trials needed to achieve the desired confidence,

$$Pr[|p - \hat{p}_i| > \frac{1}{4}] < 2e^{-2n \frac{1}{16}} < \delta$$

$$n \geq 8 \ln \frac{2}{\delta}$$